======= EPODOC =======

TI - METHOD FOR CONVERTING INPUT PROGRAM OF TRANSLATOR AND DEVICE WHICH IMPLEMENTS SAID METHOD

AB - FIELD: translators. SUBSTANCE: syntactic parser stores pointers in table of identifiers for syntactic tree instead of storing syntactic tree tokens. Table of identifiers has current effective definition for each identifier, which alters during input and output of separate units of input program of translator. Current effective definition always points to position in symbol table where information about current definition of identifier is stored. Syntactic parser of translator uses information which is stored in syntactic tree in order to exclude search in symbol table. EFFECT: increased functional capabilities. 13 cl, 7 dwgt

PN - RU2103728 C 19980127

AP - RU19950118250 19951024

PR - RU19950118250 19951024

PA - SAFONOV VLADIMIR OLEGOVICH

IN - SAFONOV VLADIMIR OLEGOVICH

DT - I

======= WPI =======

TI - Conversion of input programme of translator - includes input of input programme of translator and generation of syntax tree for input programme in memory having indicators for new elements of identifiers table

AB - RU2103728 Conversion of an input programme includes use of a data processing system with a memory in which the input programme is passed to a translator, a syntax tree unit is generated for an input programme in the memory and indicators are recorded in the syntax tree unit for a table of identifiers in the memory containing an indicator of elements of the list table for the identifiers in the input programme.

- Semantic analysis is carried out of the input programme using the indicators recorded in the syntax tree unit and the current actual determination is always indicated at the position of symbols in the table where information is stored of current determination of an identifier. Data recorded in the tree are used to eliminate the requirement to search for symbols in the table.
- USE - Realisation of tables of coded symbols ensuring rapid access to identifiers of tables.
- ADVANTAGE - Improvement organisation of access to conversion tables.
- (Dwg.0/7)

PN - RU2103728 C1 19980127 DW199837 G06F9/455 013pp

PR - RU19950118250 19951024

PA - (SAFO-I) SAFONOV V O

IN - SAFONOV V O

MC - T01-F05A T01-F05G3 T01-J20B T01-S01B

DC - T01

IC - G06F9/44 ;G06F9/455

AN - 1998-435768 [37]

(19) RU (11) 2103728 (13) C1

(51) 6 G 06 F 9/455, 9/44

Комитет Российской Федерации
по патентам и товарным знакам

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ
к патенту Российской Федерации

1

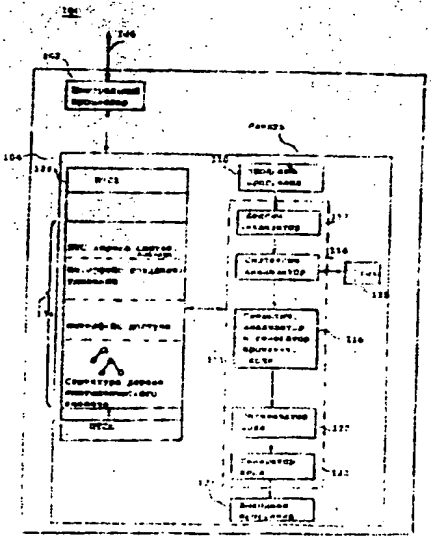
(21) 95118250/09 (22) 24.10.95
(46) 27.01.98 Бюл. № 3
(76) Сафонов Владимир Олегович
(56) EP. патент, 0372834. кл. G 06 F 9/455, 1989.

(54) СПОСОБ ПРЕОБРАЗОВАНИЯ ВХОДНОЙ ПРОГРАММЫ ТРАНСЛЯТОРА И УСТРОЙСТВО ДЛЯ ЕГО ОСУЩЕСТВЛЕНИЯ

(57) Предложены способ и устройство для преобразования входной программы транслятора в программу на его выходном языке. Синтаксический анализатор запоминает указатели таблицы идентификаторов для синтаксического дерева вместо запоминания

2

имен в синтаксическом дереве. Таблица идентификаторов имеет текущее действующее определение для каждого идентификатора, которое изменяется при вводе и выводе отдельных блоков входной программы транслятора. Текущее действующее определение всегда указывает на местоположение в таблице символов, где хранится информация о текущем определении идентификатора. Синтаксический анализатор в трансляторе использует информацию, запомненную в синтаксическом дереве для исключения необходимости поиска в таблице символов. 4 с. и 8 з. п. ф-лы, 7 ил.



RU 2103728 C1
RU 2103728 C1

Изобретение относится к транслятору для машинного языка программирования высокого уровня, в частности к способу и устройству для реализации таблицы кодировки символов, которая обеспечивает быстрый доступ к идентификаторам таблицы кодировки символов.

Транслятор представляет собой машинную программу, которая преобразует программу на входном языке транслятора, составленную на машинном языке высокого уровня, легко воспринимаемом людьми, в программу на выходном языке транслятора, исполняемую вычислительной машиной. В типовом случае транслятор включает несколько функциональных блоков. Например, обычный транслятор может включать лексический анализатор, который осуществляет просмотр в исходной программе транслятора и идентифицирует последовательные лексемы (минимальные единицы языка, имеющие значения) в исходной программе.

Обычный транслятор также включает синтаксический анализатор, который получает в качестве входных данных грамматику, определяющую язык, подлежащий транслированию, и последовательность операций, связанных с грамматикой. Синтаксический анализатор образует "синтаксическое дерево" (дерево грамматического разбора) для операторов исходной программы в соответствии с грамматическими правилами и операциями. Для каждого оператора исходной программы синтаксический анализатор генерирует синтаксическое дерево исходного ввода данных рекурсивным способом "снизу вверх" в соответствии с подходящими грамматическими правилами и операциями. Таким образом, синтаксическое дерево формируется из узлов, соответствующих одному или более грамматическим правилам. Генерация синтаксического дерева позволяет синтаксическому анализатору определять, подчиняются ли части входной программы правилам грамматики. Если нет, то синтаксический анализатор генерирует сообщение об ошибке. Таким образом, синтаксический анализатор выполняет синтаксическую проверку, но обычно не проверяет содержание ("семантику") исходной программы. Примером известных средств синтаксического анализа может служить синтаксический анализатор LALR (lookahead, left right - "с просмотром вперед, слева направо"), который описан в главе 4 монографии Aho, Sethi, Ullman, "Compiliers: Principles, Techniques and Tools".

Обычные синтаксические анализаторы также создают "таблицу перечня имен" (NL-таблицу), также называемую таблицей кодировки символов, которая отслеживает информацию, касающуюся каждого идентификатора, определенного в исходной программе. Эта информация включает имя и тип каждого идентификатора, его класс (переменная, постоянная, процедура и т.п.), уровень вложенности блока, где он описан, и другую информацию, более специфическую для класса.

В обычных трансляторах после того как исходная программа синтаксически проанализирована, она вводится в семантический анализатор, который осуществляет проверку на наличие семантических ошибок, таких как несоответствие типов и т.п. Семантический анализатор обращается к таблице перечня имен для осуществления семантической проверки с использованием идентификаторов. После семантической проверки транслятор генерирует промежуточный код, оптимизирует промежуточный код и генерирует программу на выходном языке.

Важно, чтобы транслятор выполнял свои функции быстро и эффективно. В обычных трансляторах построение семантического анализатора обусловливает недостаточную эффективность его функционирования. В частности, в обычных трансляторах семантический анализатор должен выполнять просмотр таблицы перечня имен всякий раз, когда он осуществляет проверку семантики с использованием идентификатора. Линейный поиск в таблице перечня имен требует в среднем N/2 шагов, где N - общее число идентификаторов в NL-таблице. В других известных трансляторах таблица перечня имен рандомизируется (хэшируется) в H перечней, что дает в результате в среднем N/(2*H) операций поиска. Поиск в таблице перечня имен представляет собой широко распространенную операцию семантического анализа; вследствие чего обеспечение повышения скорости доступа к компонентам таблицы перечня имен привело бы в повышению эффективности функционирования транслятора в целом.

Задача изобретения - преодоление проблем, а устранение недостатков технических решений, известных из предшествующего уровня техники, и улучшение организации доступа и способа обращения к таблице перечня имен.

Вместо запоминания имени идентификатора в синтаксическом дереве и затем поиска в таблице перечня имен при семантическом анализе синтаксический анализатор запоминает указатель (ссылку) на элемент таблицы

идентификаторов (описаний элементов). Эта таблица содержит символы идентификаторов (каждый идентификатор имеет однозначно определенный элемент таблицы идентификаторов). Элемент таблицы идентификаторов содержит поле текущего действующего определения (CED-поле), которое указывает на текущее определение для идентификатора в таблице имен. Таблица идентификаторов создается лексическим анализатором, когда он пропускает лексемы на синтаксический анализатор.

Для каждого блока в исходной программе изобретение обеспечивает поддержание "локального перечня определений" для идентификаторов, которые были определены в пределах блока. Изобретение обеспечивает модифицирование CED-полей таблицы идентификаторов, когда блоки вводятся и выводятся и когда идентификаторы определяются так, чтобы CDE-поля указывали на текущее определение каждой переменной. Дисплейный файл отслеживает идентификаторы, определенные в каждом блоке. Текущий блок или уровень исходной программы называется "контекстом" исходной программы.

После того как таблица идентификаторов и таблица перечня имен созданы и указатели для таблицы идентификаторов запомнены в синтаксическом дереве, семантический анализатор может обратиться к таблице перечня имен без осуществления какого-либо поиска. Семантический анализатор просто использует указатель для таблицы идентификаторов для получения указателя текущего действующего определения для таблицы перечня имен. Дисплейный файл и локальный перечень определений для каждого блока обеспечивают, чтобы указатель текущего действующего определения в таблице идентификаторов указывал на текущее определение для каждого идентификатора.

В предпочтительном варианте указатель для таблицы идентификаторов запоминается в синтаксическом дереве.

Поставленная задача решена в способе преобразования входной программы транслятора, основанном на использовании системы обработки данных с памятью, в котором согласно изобретению осуществляют ввод входной программы транслятора, генерацию узла синтаксического дерева для входной программы в памяти, запоминание в узле синтаксического дерева указателя для таблицы идентификаторов в памяти, содержащей указатель элемента таблицы перечня имен для идентификатора во входной программе, выполнение семантического анализа входной программы с использованием указателя запомненного в узле синтаксического дерева.

Цели и преимущества изобретения будут понятны из нижеследующего описания и при его осуществлении будут достигнуты с помощью признаков изобретения и их комбинаций, охарактеризованных в пунктах формулы изобретения.

На фиг. 1 приведена блок-схема компьютерной системы, соответствующей предпочтительному варианту осуществления изобретения; на фиг. 2 - формат структур данных, запомненных в памяти, показанной на фиг. 1, при этом структура данных включает таблицу идентификаторов, таблицу перечня имен и синтаксическое дерево; на фиг. 3,а и 3,б - блок-схемы алгоритмов операций, выполняемых для создания структур данных, показанных на фиг. 2; на фиг. 4 - блок-схема алгоритма, иллюстрирующая дополнительные операции, выполняемые при создании структур данных, показанных на фиг. 2; на фиг. 5 - пример содержания структуры данных (фиг. 2) при создании синтаксического дерева; на фиг. 6 - другой пример содержания структуры данных (фиг. 2) при создании синтаксического дерева; на фиг. 7 - блок-схема алгоритма, иллюстрирующая операции, выполняемые при семантическом анализе.

Ниже будут описаны предпочтительные варианты осуществления изобретения. Одинаковые элементы, насколько это возможно, обозначены одинаковыми ссылочными позициями.

На фиг. 1 представлена блок-схема компьютерной системы 100, выполненной согласно изобретению. Компьютерная система 100 включает центральный процессор 102, память 104, шины ввода/вывода 106. Специалистам в данной области техники должно быть ясно, что компьютерная система 100 может также включать в свой состав множество других элементов, таких как дисковые запоминающие устройства, клавиатуры, соединения с сетями, дополнительные шины памяти, дополнительные центральные процессоры и т.д.

Память 104 включает входную программу 110 транслятора, транслятор 111 и выходную программу 124 транслятора. Транслятор 111 включает лексический анализатор 112, синтаксический анализатор 114, семантический анализатор 116, оптимизатор кода 120 (не обязателен), генератор кода 122. Транслятор 111 вводит входную программу 110 и обрабатывает входную программу, преобразуя ее в выходную программу 124. Элементы 112-116 транслятора обрабатывают каждый

оператор входной программы 110 для генерации промежуточного кода.

Специалистам в данной области техники должно быть ясно, что все элементы транслятора 111 реализованы в виде команд, запомненных в памяти 104 и выполняемых центральным процессором 102. Синтаксический анализатор 114 генерирует синтаксическое дерево 133 и обращается к нему с использованием функций доступа 134 к синтаксическому дереву. Семантический анализатор 116 рекурсивно проходит синтаксическое дерево 133 для всей входной программы, созданной синтаксическим анализатором 114, вызывая соответствующие подпрограммы семантического анализа.

Семантический анализатор 116 распечатывает сообщения об ошибках и/или останавливает выполнение программы, если обнаружены семантические ошибки. Если семантических ошибок не обнаружено, то семантический анализатор 116 выводит деревья программно-совместимых компьютерных систем (ПСК) способом, хорошо известным специалистам в данной области техники. Генератор промежуточного кода блока 116 транслирует ПСК-деревья в представление для предварительной обработки данных также известным способом (Aho et al.,"Compilers: Principles, Techniques and Tools", p.p. 735-737). В первом предпочтительном варианте выполнения изобретения (многопроходной транслятор) синтаксический и семантический анализ выполняются раздельно на соответствующих проходах. Во втором предпочтительном варианте осуществления изобретения выполнение синтаксического и семантического анализа осуществляется смешанным образом.

Оптимизатор кода 120 (используемый дополнительно) оптимизирует это представление и выводит представление для предварительной обработки дачных в оптимизированной форме (например, неиспользованные фрагменты кода могут быть опущены). Генератор кода 122 предпочтительно транслирует представление для предварительной обработки данных в выходную программу 124 способом, хорошо известным специалистам в данной области техники.

Память 104, таким образом, содержит структуры данных, включающие таблицу перечня имен, таблицу идентификаторов и множество локальных перечней определений, а также общий дисплейный файл; каждый из этих элементов будет подробно рассмотрен ниже.

На фиг. 2 показан формат структур данных 130, запомненных в памяти, показанной на фиг. 1, где структуры данных включают таблицу идентификаторов 202, таблицу перечня имен 204 и синтаксическое дерево 133. Эти структуры данных обеспечивают возможность семантическому анализатору 116 ссылаться на идентификационную информацию из таблицы перечня имен без выполнения поиска в таблице перечня имен. Синтаксическое дерево 133 генерируется для следующего исходного кода 201:

```
VAR a: целое число
...
a : = 0
```

Таким образом, узел 210 синтаксического дерева 133 соответствует переменной "a". Узел 210 включает указатель 212, который указывает на элемент таблицы идентификаторов 220 для идентификатора "a". Имя поля указателя 212 в узле 210 синтаксического дерева есть "CPTR".

Ниже описаны форматы таблицы идентификаторов 202 и таблицы перечня имен 204. Таблица идентификаторов 202 включает элемент для каждого идентификатора, используемого во входной программе 110. Каждый элемент в таблице идентификаторов 202 предпочтительно имеет два поля: поле идентификатора 222, содержащее имя идентификатора (например "a"), и поле ID_NL 224, содержащее указатель для текущего действующего определения (CED) идентификатора в таблице перечня имен. Значение поля ID_NL 224 может изменяться в зависимости от текущего блока входной программы, которая подлежит трансляции.

Таблица перечня имен 204 включает элемент для каждого идентификатора, определяемого в каждом блоке входной программы 110. Каждый элемент 230 в таблице перечня имен 204 предпочтительно имеет пять полей: 1) поле 232, индицирующее, является ли идентификатор переменной ("VAR") или постоянной; 2) поле NL_ID 234, содержащее обратный указатель для соответствующего элемента 220 в таблице идентификатора 202; 3) поле NL_PREV 236, содержащее указатель для предыдущего определения идентификатора во внешнем блоке (если есть) или NIL-указатель; 4) поле NL_NXTLOC 238, содержащее указатель для следующего элемента в "локальном перечне определений" для текущего блока; 5) поле 239 типа, индицирующее тип идентификатора (например, "целое число"). Специалистам в данной области техники должно быть ясно, что элемент 230 в таблице перечня имен 204 может также иметь другие поля, содержащие информацию об идентификаторах. Эти поля не рассматриваются здесь в

целях сохранения наглядности описания примеров.

На фиг. 3 - 7 показан пример создания и использования структур данных (фиг. 2) в процессе трансляции. Следует иметь в виду, что операции, иллюстрируемые на фиг. 3, 4 и 7, реализованы как компьютерные команды, запомненные в памяти 104 и выполняемые центральным процессором 102. В рассматриваемом примере использована следующая входная программа транслятора:

```
программа p;
var a: целое число
b: действит.; /*1*/
процедура q;
var a: символ
b: логическое
начало /*2*/
a:= "*";
b:= "истина"
конец; /*3*/
начало
a:=0; b:1.0;
конец.
```

В следующем примере показаны структуры данных в различных точках 1, 2 и 3, как показано в комментариях программы p выше.

Фиг. 3,а, 3,б и 4 иллюстрируют операции, выполняемые при создании структур данных, представленных на фиг. 5 и 6. В предпочтительном варианте выполнения операции, показанные на фиг. 3,а, выполняются всякий раз, когда вводится программный блок. Аналогично в предпочтительном варианте выполнения операции, показанные на фиг. 3,б, выполняются, когда определен идентификатор. В многопроходном трансляторе, соответствующем первому варианту осуществления изобретения, операции выполняются синтаксическим анализатором 114. Во втором предпочтительном варианте операции выполняются семантическим анализатором, как показано ниже.

На фиг. 5 и 6 показан дисплейный файл 240. Каждый элемент 250 дисплейного файла 240 соответствует блоку (также называемому "уровнем") входной программы 110 транслятора. В рассматриваемом примере программа p представляет первый уровень, а процедура q - второй уровень. Каждый элемент в дисплейном файле 240 указывает на первый элемент и последний элемент з локальном перечне определений для блока. ПЕРВЫЙ указывает на первый элемент в локальном перечне определений, а ПОСЛЕДНИЙ указывает на последний элемент в локальном перечне определений. Каждый локальный перечень определений сформирован из элементов, связанных полями NL_NXTLOC 228 таблицы 204 перечня имен.

Структуры данных по фиг. 5 и 6 первоначально создаются синтаксическим анализатором 114 во время создания им синтаксического дерева 133. На фиг. 3,а, когда синтаксический анализатор 114 вводит новый блок, уровень ("L") получает приращение (например, на "один" "1") на этапе 304 (фиг. 3,а). Когда это создается впервые, то дисплейный файл 240 - пустой. Новый локальный перечень определений добавляется для каждого нового блока, вводимого на этапе 306. Каждый локальный перечень определений сначала не заполнен. Элементы добавляются к локальному перечню определений по мере того, как идентификаторы определяются в соответствующем блоке.

На фиг. 4 представлена блок-схема алгоритма, иллюстрирующего дополнительные операции, выполняемые для создания структур данных по фиг. 2. Операции, показанные на фиг. 4, выполняются, когда идентификатор определен. Когда синтаксический анализатор 114 обнаружит новое определение идентификатора (операция 402), синтаксический анализатор выполняет операции согласно фиг. 4 для создания нового элемента таблицы перечня имен для идентификатора. Синтаксический анализатор затем добавляет идентификатор к локальному перечню определений для текущего блока.

В следующем примере предполагается, что синтаксический анализатор создает элемент таблицы перечня имен для целочисленной переменной "a" программы p. На этапе 404 синтаксический анализатор создает элемент 230 таблицы перечня имен для "a" и запоминает индикацию того, что "a" есть переменная, в поле 232. Синтаксический анализатор 114 также определяет указатель для "a", который идентифицирует соответствующий элемент в таблице идентификаторов 202. Здесь не имеется элемента в таблице идентификаторов 202 для "a", так что элемент 220 создается на этапе 408.

На этапе 410, поскольку не имелось элемента в таблице идентификаторов 202 для "a", поле NL-PREV устанавливается на NIL. Если бы имелся элемент для "a" в таблице идентификаторов 202, то поле NL_PREV устанавливалось бы на элемент таблицы перечня имен для "a" на этапе 414. Переменное имя "a" запоминается в поле 222 на этапе 412. На этапе 416 указатель 224 ID_NL установлен для указания на новый элемент 230 таблицы перечня имен.

На этапе 418 поле NL_ID 234 устанавливается для указания на новый элемент 220

таблицы идентификаторов. На этапе 420 элемент 230 таблицы перечня имен добавляется к локальному перечню определений и поля ПЕРВЫЙ и ПОСЛЕДНИЙ текущего элемента дисплейного файла 240 устанавливаются для указания на первый и последний поля в локальном перечне определений. На этапе 422 осуществляется возврат указателя к вновь созданному полю 220 таблицы идентификаторов. Возвращенный указатель запоминается в поле 212 соответствующего узла синтаксического дерева.

Когда синтаксический анализатор 114 обнаруживает ранее определенный идентификатор, он запоминает указатель для соответствующего элемента таблицы идентификаторов в поле 212 узла синтаксического дерева (вместо запоминания имени идентификатора).

На фиг. 5 показана структура данных в точке /*1*/ программы p после определения целочисленного идентификатора "a" и действительного идентификатора "b". Указатель ПЕРВЫЙ элемента 250 дисплейного файла указывает на элемент 230 таблицы перечня имен для "a". Указатель ПОСЛЕДНИЙ указывает на элемент 251 таблицы перечня имен для "b". На фиг. 6 показаны структуры данных в точке /*2*/ в процедуре q после определения символьного идентификатора "a" и логического идентификатора "b". Как показано на фиг. 6, после ввода процедуры q текущий уровень ("L") становится "2", и новый локальный перечень определений (указанный элементом 251) инициализируется для процедуры q. После того как этапы (фиг. 4) выполнены для определений новых идентификаторов "a" и "b" в процедуре q, элементы 220 и 221 в таблице идентификаторов 202 указывают на новые текущие действующие определения для "a" и "b". Таким образом, например, в точке /*2*/ в процедуре q синтаксический анализ определения идентификатора "a" модифицирует элемент 220 в таблице идентификаторов 202 для указания на элемент 260 в таблице перечня имен 206. Аналогично синтаксический анализ отсылки на идентификатор "b" приведет к модифицированию элемента 221 таблицы идентификаторов 202 для указания на элемент 261 таблицы перечня имен 204.

В точке /*2*/ программы p текущий элемент 260 таблицы перечня имен для "a" указывает вновь на элемент 230 более высокого уровня для "a". Аналогично текущий элемент 261 таблицы перечня имен для "b" указывает на элемент 231 более высокого уровня для "b".

В точке /*3*/ программы p синтаксический анализатор только что обработал код входной программы, который выходит из блока процедуры q, и выполняются операции по фиг. 3,б. На этапе 354 для каждого элемента локального перечня определений (определен указателями 251 ПЕРВЫЙ и ПОСЛЕДНИЙ) каждое поле ID_NL 224 соответствующей таблицы идентификаторов устанавливается для указания либо на вариант более высокого блока для переменной (как индицируется полем NL_PREV 236) или на NIL. Этап 356 понижает указатель текущего уровня ("L") для указания на элемент 250. Эти операции обеспечивают возврат структур данных к состоянию, показанному на фиг. 5.

Таблица идентификаторов 202, таблица перечня имен 204, дисплейный файл 240 и множество локальных перечней определений создаются синтаксическим анализатором 114. Семантический анализатор 116 использует структуры данных при семантическом анализе для быстрого доступа к таблице перечня имен. При выполнении своих операций семантический анализатор 116 устанавливает указатель уровня L для дисплейного файла 240 и поля ID_NL 224 таблицы идентификаторов 202, когда семантический анализатор 116 осуществляет ввод и вывод блоков данных. Нет необходимости создавать структуры данных повторно в ходе семантического анализа, поскольку данные, созданные синтаксическим анализатором, сохраняются в структурах данных.

На фиг. 7 представлена блок-схема алгоритма, иллюстрирующая операции, выполняемые при семантическом анализе, в первом варианте осуществления изобретения. Фиг. 7 иллюстрирует функционирование многопроходного транслятора, в котором семантический анализ выполняется после синтаксического анализа. Когда семантический анализатор 116 вводит новый блок входной программы транслятора на этапе 702, уровень дисплейного файла (L) повышается на этапе 704 для указания на другой локальный перечень определений (как и образован полями NL_NXTLOC) таблицы перечня имен 204. Для каждого элемента локального перечня определений соответствующее поле ID_NL 224 таблицы идентификаторов 202 устанавливается для указания на элемент локального перечня определений на этапе 706. В результате осуществления операций на этапах 702-706 соответствующие поля ID_NL устанавливаются для указания на элементы таблицы перечня имен для

идентификаторов, вновь определенных в блоке данных.

Когда семантический анализатор 116 обнаруживает идентификатор в синтаксическом дереве 133 (этап 708), он использует поле 212 синтаксического дерева для указания на таблицу идентификаторов 202 для получения элемента таблицы перечня имен для идентификатора (этап 710). Поле ID_NL 224 в таблице идентификаторов 202 указывает на текущее действующее определение в таблице перечня имен 204 для идентификатора. Таким образом, семантический анализатор 116 не должен осуществлять поиск в таблице перечня имен 204 для нахождения имени идентификатора. Более того, использование поля ID_NL 224 в таблице идентификаторов 202 обеспечивает то, что семантический анализатор 116 получает доступ к корректной версии идентификатора, если идентификатор определен несколько раз в различных блоках.

Когда семантический анализатор 116 выводит блок входной программы на этапе 714 для каждого элемента в текущем локальном перечне определений, соответствующее поле ID_NL 224 устанавливается на этапе 716 для указания на элемент более высокого блока (или на NIL), как описано выше со ссылками на фиг. 3,б. Уровень (L) дисплейного файла понижается на этапе 718. В результате осуществления операций на этапах 714-718 соответствующие поля ID_NL уже не указывают на идентификаторы, определенные в выведенном блоке.

Во втором предпочтительном варианте осуществления изобретения транслятор не выполняет синтаксической и семантической проверок в отдельных проходах. Вместо этого по меньшей мере часть синтаксического и семантического анализа перемешана и поля CED (текущего действующего определения) устанавливаются или восстанавливаются только один раз на блок исходного кода. В этом варианте, например, обработка идентификатора "a" осуществляется следующим образом.

Синтаксический анализатор 114 анализирует синтаксически определение для "a" и создает его узел синтаксического дерева. Семантический анализатор 116 создает новый элемент перечня имен для "a" и вводит этот элемент перечня имен в текущий контекст (т. е. устанавливает поле CED идентификатора id(a) для указания на N и обновляет соответствующее поле NL_PREV для указания на предыдущий контекст "a", если он есть). Семантический анализ затем выполняется над определением "a". Установка поля CED в соответствии с новым контекстом "a" сохраняется неизменной до конца блока или до получения и вого определения "a".

Когда во втором варианте синтаксический анализатор 114 достигает присвоенного определения "a", контекст для "a" уже установлен посредством семантического анализа определения, как описано выше. Семантический анализатор 116 использует указатель синтаксического дерева и поле CED для того, чтобы избежать необходимости поиска в таблице перечня имен для "a" при семантическом анализе присвоенного определения "a". Во втором варианте в конце процедуры или блока семантический анализатор 116 восстанавливает все старые отсылки CED локальных идентификаторов .

Таким образом, во втором варианте этапы 304 и 306 (фиг. 3,а), этапы 354 и 356 (фиг. 3,б) и этапы 404-422 (фиг. 4) выполняются семантическим анализатором 116, работающим во взаимосвязи с синтаксическим анализатором 114. Так как семантический анализатор 116 не образует часть отдельного прохода, этапы, показанные на фиг. 7, которые определяют, когда блок введен и выведен, не требуются во втором варианте.

В одном из практических примеров осуществления второго варианта подпрограммы семантического анализа обрабатывающие указатели CED представляют собой часть семантических подпрограмм, включенных в качестве операций в грамматику транслируемого языка. Эта реализация рассмотрена в совместно поданной заявке того же заявителя на "Способ и устройство для эффективной оценки семантических признаков при синтаксическом анализе с просмотром вперед, слева направо."

Другие варианты осуществления изобретения могут быть очевидны для специалистов в данной области техники на основе описания изобретения и примеров его использования.

Кроме того, для специалистов в данной области техники должно быть ясно, что настоящее изобретение может быть использовано совместно с транслятором для любого языка высокого уровня, использующего таблицу перечня имен (таблицу кодировки символов). Хотя изобретение было описано в связи с обычным семантическим анализатором, ясн , что оно может быть использовано для транслятора, использующего семантические признаки, как описано в вышеупомянутой совместно поданной заявке.

Следует иметь в виду, что описание и приведенные в нем варианты осуществления изобретения должны рассматриваться только как возможные примеры, в то время как действительный объем изобретения должен определяться пунктами формулы изобретения.

ФОРМУЛА ИЗОБРЕТЕНИЯ

1. Способ преобразования входной программы транслятора, основанный на осуществлении операций системой обработки информации, имеющей память, ***отличающийся*** тем, что включает операции ввода входной программы транслятора, генерации узла синтаксического дерева для входной программы в памяти, запоминания в узле синтаксического дерева указателя для таблицы идентификаторов в памяти, которая содержит указатель элемента таблицы перечня имен для идентификатора входной программы, выполнения семантического анализа входной программы с использованием указателя, запомненного в узле синтаксического дерева для обращения к элементу таблицы перечня имен для идентификатора.

2. Способ формирования синтаксического дерева посредством транслятора, осуществляющего ввод входной программы, основанный на осуществлении операций системой обработки информации, имеющей память, ***отличающийся*** тем, что включает операции обнаружения ввода нового блока входной программы, формирования нового локального перечня определений для нового блока в памяти, обнаружения определения идентификатора во входной программе, добавления нового элемента к таблице перечня имен в памяти для идентификатора, формирования в памяти узла синтаксического дерева, имеющего косвенный указатель для нового элемента таблицы перечня имен, и добавления нового элемента таблицы перечня имен к новому локальному перечню определений.

3. Способ по п.2, ***отличающийся*** тем, что дополнительно включает операции обнаружения вывода блока входной программы и понижения индикатора уровня так, что новый локальный перечень определений был недействующим.

4. Способ по п.2, ***отличающийся*** тем, что дополнительно включает операции формирования нового элемента таблицы идентификаторов, включающего указатель текущего действующего определения, который указывает на новый элемент таблицы перечня имен, причем операция формирования узла синтаксического дерева включает формирование узла синтаксического дерева, имеющего указатель для нового элемента таблицы идентификаторов.

5. Способ по п.4, ***отличающийся*** тем, что дополнительно включает операцию изменения текущего указателя действующего определения в существующем элементе таблицы идентификаторов для указания на новую таблицу перечня имен при вводе блока.

6. Способ по п.1, ***отличающийся*** тем, что операция запоминания осуществляется синтаксическим анализатором в многопроходном трансляторе.

7. Способ по п.1, ***отличающийся*** тем, что операции запоминания и выполнения семантического анализа осуществляются семантическим анализатором транслятора.

8. Устройство для преобразования входной программы транслятора, ***отличающееся*** тем, что содержит память, предназначенную для запоминания по меньшей мере входной программы транслятора, таблицы идентификаторов и таблицы перечня имен, содержащий элемент таблицы перечня имен, генератор синтаксического дерева, предназначенный для генерации узла синтаксического дерева в памяти для входной программы, блок указателя, предназначенный для запоминания в узле синтаксического дерева указателя для таблицы идентификаторов в памяти, которая содержит указатель элемента таблицы перечня имен для идентификатора во входной программе, и семантический анализатор, предназначенный для выполнения семантического анализа входной программы с использованием указателя, запомненного в узле синтаксического дерева.

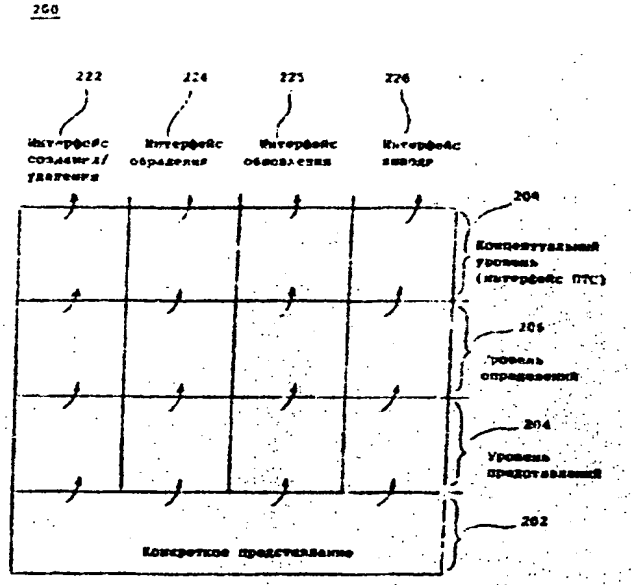
9. Транслятор, предназначенный для формирования синтаксического дерева в памяти для входной программы транслятора, хранящейся в памяти, ***отличающийся*** тем, что содержит первый блок синтаксического анализатора, предназначенный для обнаружения ввода нового блока входной программы, второй блок синтаксического анализатора, предназначенный для формирования нового локального перечня определений для нового блока в память, третий блок синтаксического анализатора, предназначенный для обнаружения определения идентификатора во входной программе, четвертый блок синтаксического анализатора, предназначенный для добавления нового элемента в таблицу перечня имен в памяти для

идентификаторов, пятый блок синтаксического анализатора, предназначенный для формирования в памяти узла синтаксического дерева, содержащего косвенный указатель на новый элемент в таблице перечня имен, и шестой блок синтаксического анализатора, предназначенный для добавления нового элемента таблицы перечня имен к новому локальному перечню определений.

10. Транслятор по п.9, ***отличающийся*** тем, что дополнительно содержит седьмой блок синтаксического анализатора, предназначенный для обнаружения ввода блока входной программы и для понижения индикатора уровня так, чтобы новый локальный перечень определения был недействующим.

11. Транслятор по п.9, ***отличающийся*** тем, что дополнительно содержит седьмой блок синтаксического анализатора, предназначенный для формирования нового элемента таблицы идентификаторов, включающего указатель текущего действующего определения, который указывает на новый элемент таблицы перечня имен, причем пятый блок синтаксического анализатора включает часть, предназначенную для формирования узла синтаксического дерева, содержащего указатель нового элемента таблицы идентификаторов.

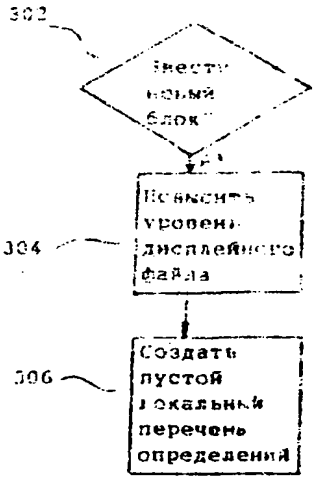
12. Транслятор по п.9, ***отличающийся*** тем, что дополнительно содержит седьмой блок синтаксического анализатора, предназначенный для изменения текущего указателя действующего определения в существующем элементе таблицы идентификаторов для указания на новую таблицу перечня имен при выводе блока входной программы.



Фиг. 2

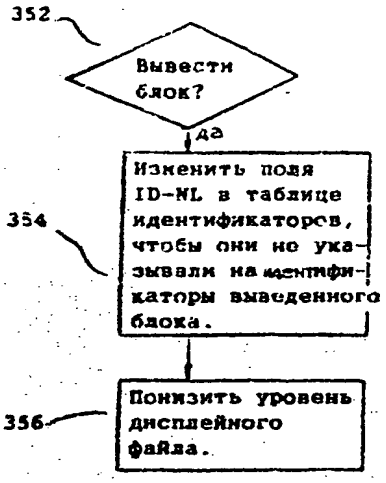
300

а)

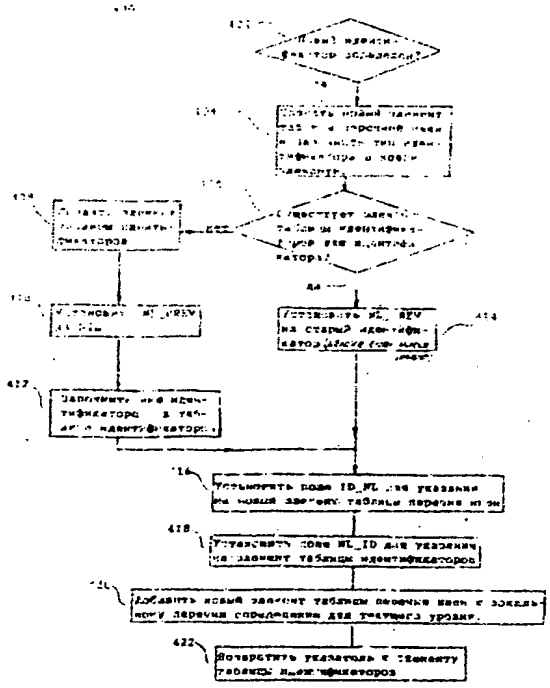


350

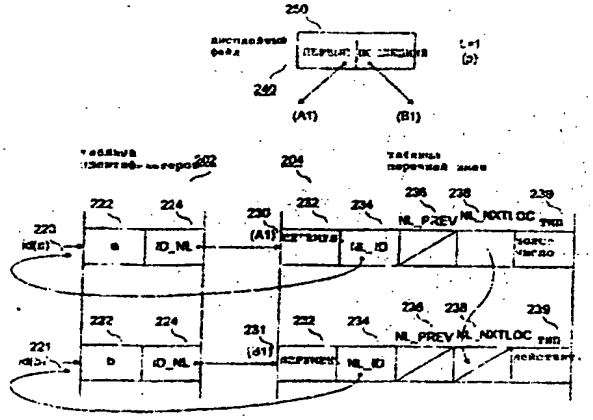
б)



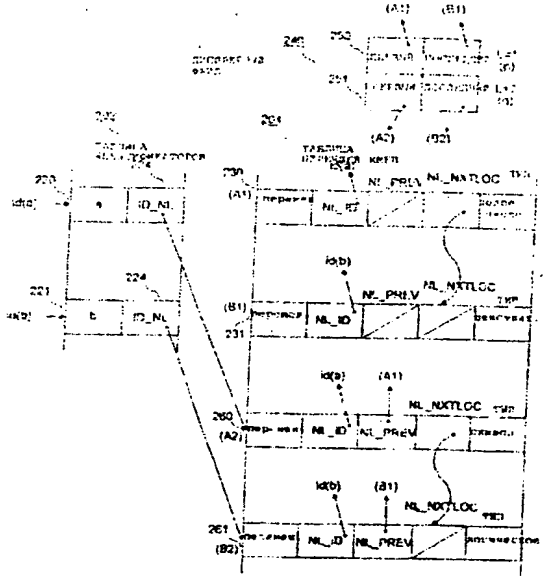
Фиг.3



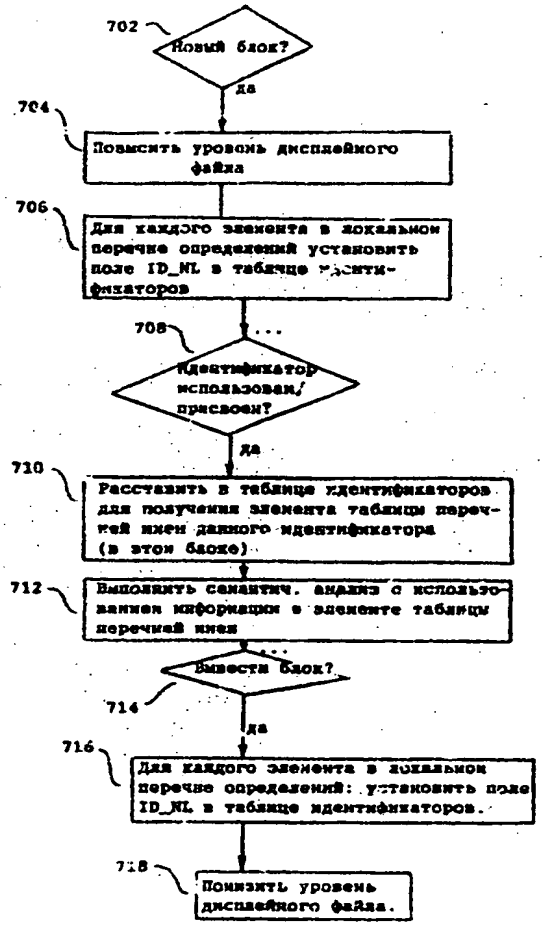
Фиг. 4



Фиг. 5



фиг. 6



Использовать элемент перечня имен

Фиг.7

Заказ 3и Подписное
ВНИИПИ. Рег. ЛР № 040720
113834, ГСП, Москва, Раушская наб.,4/5

121873, Москва, Бережк вская наб., 24 стр. 2.
Производственное предприятие «Патент»